What is claimed is:

1. A file server comprising storage containing a file system, and a processor coupled to the storage for accessing the file system, wherein the file system includes a production file, read-only snapshot copies of the production file, and at least one read-write snapshot copy of the production file, wherein the production file and the snapshot copies of the production file are organized as a version set including an inode for the production file and an inode for each snapshot copy of the production file, and a set of file blocks including data blocks and indirect blocks that are shared among the production file and the snapshot copies of the production file.

2. The file server as claimed in claim 1, which is programmed to create a new fully preallocated production file by allocating an inode for the production file, allocating to the production file data blocks for a specified size of the production file, and allocating to the production file any and all indirect blocks needed to link the data blocks of the production file to the inode of the production file.

3. The file server as claimed in claim 1, which is programmed to create a new sparse production file by initially allocating just an inode for the production file, and wherein the file server is programmed to allocate other blocks for the sparse production file as needed when data blocks of the sparse production file are written to.

4. The file server as claimed in claim 1, wherein the version set includes a version chain linking the inode for the production file to the nodes for the read-only snapshot copies of the production file.

5. The file server as claimed in claim 4, wherein the inode for each read-write snapshot copy of the production file is linked to a corresponding one of the nodes for the read-only snapshot copies of the production file.

6. The file server as claimed in claim 1, wherein each inode in the version set includes a version pointer field and a branch pointer field, contents of the version pointer fields link the nodes of the read-only snapshot copies of the production file into the version set, and contents of the branch pointer fields link each inode of each read-write snapshot copy of the production file into the version set.

7. The file server as claimed in claim 6, wherein the contents of the version pointer fields form a version chain linking the inode of the production file to nodes of the read-only copies of the production file.

8. The file server as claimed in claim 7, wherein the contents of the branch pointer fields link each inode of each read-write snapshot copy of the production file system to an inode of a respective one of the read-only snapshot copies of the production file.

9. The file server as claimed in claim 1, wherein the file server is programmed for parsing path names through the version set to locate the nodes in the version set, each path name including a name for the production file, and each path name for a snapshot copy of the production file includes the name for the production file followed by a delimiter symbol followed by a name for the snapshot copy of the production file.

10. The file server as claimed in claim 9, wherein each path name for a read-write snapshot copy of the production file includes the name for the production file followed by a delimiter symbol followed by a name for a read-only snapshot copy of the production file followed by a delimiter symbol followed by a name for the read-write snapshot copy of the production file.

11. The file server as claimed in claim 1, wherein the file server is programmed for creating a new read-only snapshot copy of the production file, creating a new read-write snapshot copy of the production file, deleting a snapshot copy of the production file from the version set, restoring the production file with a specified snapshot copy of the production file, refreshing a specified snapshot copy of the production file, and naming the files in the version set.

12. The file server as claimed in claim 1, which is programmed for creating a new read-write snapshot copy of a specified file in the version set by allocating an inode for the new read-write snapshot copy, copying the inode of the specified file to the inode for

the new read-write snapshot copy, and designating that blocks of the specified file are shared with the new read-write snapshot copy.

13. The file server as claimed in claim 12, wherein the nodes contain pointer fields for pointers to the file blocks, and wherein the file server is programmed to designate that blocks of the specified file are shared with the new read-write snapshot copy by setting flags in the pointer fields in the inode for the read-write snapshot copy.

14. The file server as claimed in claim 1, which is programmed to maintain for each block in each read-only snapshot copy of the production file an indication of whether or not said each snapshot copy of the production file is an oldest snapshot copy of the production file including an identical version of said each block.

15. The file server as claimed in claim 14, wherein the nodes in the version set include pointer fields, and wherein each pointer field includes a pointer to a block in the version set and a flag for indicating whether or not the version of the production file of the inode is an oldest snapshot copy of the production file including an identical version of the block pointed to by the pointer in said each pointer field.

16. The file server as claimed in claim 14, wherein each inode of each read-only snapshot copy of the production file is linked to a hierarchy of blocks included in said each read-only snapshot copy of the production file, the inode of said each read-only snapshot copy of the production file includes an indication of whether or not said each

read-only snapshot copy of the production file is an oldest read-only snapshot copy of the production file including an identical version of each block that is a child of said each inode in the hierarchy of blocks included in said each read-only snapshot copy of the production file, and when said each read-only snapshot copy of the production file is not an oldest read-only snapshot copy of the production file including an identical version of said each block that is a child of said each inode in the hierarchy the of blocks, then said each read-only snapshot copy of the production file is not an oldest read-only snapshot copy of the production file including an identical version of each descendant of said each block that is a child of said each inode in the hierarchy of the blocks.

17. The file server as claimed in claim 14, which is programmed to delete a read-only snapshot copy of the production file, and when deleting the read-only snapshot copy of the production file, to keep each block for which the read-only snapshot copy is not indicated as being an oldest snapshot copy of the production file including an identical version of said each block.

18. The file server as claimed in claim 17, wherein each inode of each read-only snapshot copy of the production file is linked to a hierarchy of blocks included in said each read-only snapshot copy of the production file, and wherein the file server is further programmed to keep all descendants of said each block for which the read-only snapshot copy is not indicated as being an oldest snapshot copy of the production file including an identical version of said each block.

19. The file server as claimed in claim 14, which is programmed to delete a read-only snapshot copy of the production file, and when deleting the read-only snapshot copy of the production file, to keep each block for which the read-only snapshot copy of the production file is indicated as being an oldest snapshot copy of the production file including an identical version of said each block and a next-most recent version of the production file is indicated as not being an oldest snapshot copy of the production file including an identical version of said each block.

20. The file server as claimed in claim 19, which is further programmed, upon deleting the read-only snapshot copy of the production file, to indicate that the next-most recent snapshot copy of the production file has become an oldest snapshot copy of the production file including an identical version of said each block for which the read-only snapshot copy of the production file is indicated as being an oldest snapshot copy of the production file including an identical version of said each block and a next-most recent snapshot copy of the production file is indicated as not being an oldest snapshot copy of the production file including an identical version of said each block.

21. The file server as claimed in claim 14, which is programmed to delete a read-only snapshot copy of the production file by deallocating each block for which the read-only snapshot copy is indicated as an oldest read-only snapshot copy of the production file including an identical version of said each block and a next most recent read-only snapshot copy of the production file is also indicated as an oldest read-only snapshot copy of the production file including an identical version of a block corresponding to said

each block, wherein said each block and the block corresponding to said each block are mapped to the same logical file addresses.

22. The file server as claimed in claim 1, which is programmed for responding to a request to create a read-only snapshot copy of the production file by reserving for the production file a number of free file blocks of at least the number of blocks in the production file.

23. The file server as claimed in claim 1, which is programmed for responding to a request to create a read-write snapshot copy of a specified file in the version set by reserving for the read-write snapshot copy a number of free blocks of at least the number of blocks in the specified file in the version set.

24. The file server as claimed in claim 1, which is programmed for responding to a request to restore the production file with a specified snapshot copy of the production file by reserving for the production file a number of free blocks of at least the difference between the number of blocks in the specified snapshot copy of the production file and the number of blocks in the production file.

25. The file server as claimed in claim 1, which is programmed for restoring the production file with a specified snapshot copy of the production file by responding to a request to prepare to restore the production file by preparing to restore the production file and reporting whether or not preparation is successful, and then responding a request to

commit the preparation by restoring the production file with the specified snapshot copy of the production file.

26. The file server as claimed in claim 25, which is programmed to prepare to restore the production file by creating a read-write snapshot copy of the specified snapshot copy of the production file, and which is programmed to commit the preparation by replacing the production file with the read-write snapshot copy of the specified snapshot copy of the production file and deleting the production file, so that the read-write snapshot copy of the specified snapshot copy of the production file assumes the identity of the production file.

27. The file server as claimed in claim 1, which includes refreshing a specified snapshot copy of the production file by creating a new inode in the version set, copying contents of the inode of the specified snapshot copy into the new inode so that the new inode references blocks of the specified snapshot copy of the production file, using the inode of the specified snapshot copy to create a new snapshot copy of the production file by copying contents of the inode of the production file into the inode of the specified snapshot copy, and performing a file deletion upon the new node.

28. The file server as claimed in claim 27, which is programmed to perform the file deletion upon the new inode asynchronously after using the inode of the specified snapshot copy to create a new snapshot copy of the production file by copying contents of the inode of the production file into the inode of the specified snapshot copy.

29. A file server comprising storage containing a file system, and a processor coupled to the storage for accessing the file system, wherein the file system includes a production file, read-only snapshot copies of the production file, and at least one read-write snapshot copy of the production file, wherein the production file and the snapshot copies of the production file are organized as a version set including an inode for the production file and an inode for each snapshot copy of the production file, and a set of file blocks including data blocks and indirect blocks that are shared among the production file and the snapshot copies of the production file, wherein the file server further includes:

means for creating new read-only snapshot copies of the production file;

means for creating new read-write snapshot copies of the production file;

means for deleting a specified snapshot copy of the production file from the version set;

means for restoring the production file with a specified snapshot copy of the production file;

means for refreshing a specified snapshot copy of the production file; and

means for naming the files in the version set.

30. A file server comprising storage containing a file system, and a processor coupled to the storage for accessing the file system, wherein the file system includes a production file, and read-only snapshot copies of the production file, wherein the production file and the read-only snapshot copies of the production file are organized as a version set including an inode for the production file, an inode for each read-only snapshot copy of

the production file, and a set of file blocks including data blocks and indirect blocks that are shared among the production file and the read-only snapshot copies of the production file,

wherein the file server is programmed to maintain for each block in each snapshot copy of the production file an indication of whether or not said each snapshot copy of the production file is an oldest snapshot copy of the production file including an identical version of said each block, and

wherein the file server is programmed to delete a read-only snapshot copy of the production file, and when deleting the read-only snapshot copy of the production file, to keep each block for which the read-only snapshot copy is not indicated as being an oldest snapshot copy of the production file including an identical version of said each block.

31. The file server as claimed in claim 30, wherein each inode of each read-only snapshot copy of the production file is linked to a hierarchy of blocks included in said each read-only snapshot copy of the production file, and wherein the file server is further programmed, upon deleting the read-only snapshot copy of the production file, to keep all descendants of said each block for which the read-only snapshot copy is not indicated as being an oldest snapshot copy of the production file including an identical version of said each block.

32. The file server as claimed in claim 30, which is further programmed, upon deleting the read-only snapshot copy of the production file, to keep each block for which the read-only snapshot is indicated as being an oldest snapshot copy of the production file

including an identical version of said each block and a next-most recent snapshot copy of the production file is indicated as not being an oldest snapshot copy of the production file including an identical version of said each block.

33. The file server as claimed in claim 32, which is further programmed, upon deleting the read-only snapshot copy of the production file, to indicate that the next-most recent snapshot copy of the production file has become an oldest snapshot copy of the production file including an identical version of said each block for which the read-only snapshot is indicated as being an oldest snapshot copy of the production file including an identical version of said each block and a next-most recent snapshot copy of the production file is indicated as not being an oldest snapshot copy of the production file including an identical version of said each block.

34. The file server as claimed in claim 30, which is further programmed, upon deleting the read-only snapshot copy of the production file, to deallocate each block for which the read-only snapshot copy is indicated as an oldest snapshot copy of the production file including an identical version of said each block and a next most recent snapshot copy of the production file is also indicated as an oldest snapshot copy of the production file including an identical version of a block corresponding to said each block, wherein said each block and the block corresponding to said each block are mapped to the same logical file addresses.

35. A file server comprising storage containing a file system, and a processor coupled to the storage for accessing the file system, wherein the file system includes a production file, and snapshot copies of the production file, wherein the production file and the snapshot copies of the production file are organized as a version set including an inode for the production file, an inode for each snapshot copy of the production file, and a set of file blocks including data blocks and indirect blocks that are shared among the production file and the snapshot copies of the production file,

wherein the file server is programmed for responding to a request to create a read-only snapshot copy of the production file by reserving for the production file a number of free file blocks of at least the number of blocks in the production file.

36. The file server as claimed in claim 35, which is programmed for responding to a request to create a read-write snapshot copy of a specified file in the version set by reserving for the read-write snapshot copy a number of free blocks of at least the number of blocks in the specified file in the version set.

37. The file server as claimed in claim 35, which is programmed for responding to a request to restore the production file with a specified snapshot copy of the production file by reserving for the production file a number of free blocks of at least the difference between the number of blocks in the specified snapshot copy of the production file and the number of blocks in the production file.

38. A file server comprising storage containing a file system, and a processor coupled to the storage for accessing the file system, wherein the file system includes a production file, and snapshot copies of the production file, wherein the production file and the snapshot copies of the production file are organized as a version set including an inode for the production file, an inode for each snapshot copy of the production file, and a set of file blocks including data blocks and indirect blocks that are shared among the production file and the snapshot copies of the production file,

wherein the file server is programmed for restoring the production file with a specified snapshot copy of the production file by responding to a request to prepare to restore the production file by preparing to restore the production file and reporting whether or not preparation is successful, and then responding a request to commit the preparation by restoring the production file with the specified snapshot copy of the production file.

39. The file server as claimed in claim 38, which is programmed to prepare to restore the production file by creating a read-write snapshot copy of the specified snapshot copy of the production file, and which is programmed to commit the preparation by replacing the production file with the read-write snapshot copy of the specified read-only snapshot copy of the production file and deleting the production file, so that the read-write snapshot copy of the specified snapshot copy of the production file assumes the identity of the production file.

40. A file server comprising storage containing a file system, and a processor coupled to the storage for accessing the file system, wherein the file system includes a production file, and snapshot copies of the production file, wherein the production file and the snapshot copies of the production file are organized as a version set including an inode for the production file, an inode for each snapshot copy of the production file, and a set of file blocks including data blocks and indirect blocks that are shared among the production file and the snapshot copies of the production file,

wherein the file server is programmed for refreshing a specified snapshot copy of the production file by creating a new inode in the version set, copying contents of the inode of the specified snapshot copy into the new inode so that the new inode references blocks of the specified snapshot copy, using the inode of the specified snapshot copy to create a new snapshot copy of the production file by copying contents of the inode of the production file into the inode of the specified snapshot copy, and performing a file deletion upon the new node.

41. The file server as claimed in claim 40, wherein the file deletion upon the new inode is performed asynchronously after using the inode of the specified snapshot copy to create a new snapshot copy of the production file by copying contents of the inode of the production file into the inode of the specified snapshot copy.

42. A method of operating a file server, the file server including storage containing a file system, and a processor coupled to the storage for accessing the file system, wherein the file system includes a production file, and read-only snapshot copies of the production

file, wherein the production file and the read-only snapshot copies of the production file are organized as a version set including an inode for the production file, an inode for each read-only snapshot copy of the production file, and a set of file blocks including data blocks and indirect blocks that are shared among the production file and the read-only snapshot copies of the production file, wherein the method comprises:

maintaining for each block in each snapshot copy of the production file an indication of whether or not said each snapshot copy of the production file is an oldest snapshot copy of the production file including an identical version of said each block, and

deleting a read-only snapshot copy of the production file, wherein the deleting of the read-only snapshot copy of the production file includes keeping each block for which the read-only snapshot copy is not indicated as being an oldest snapshot copy of the production file including an identical version of said each block.

43. The method as claimed in claim 42, wherein each inode of each read-only snapshot copy of the production file is linked to a hierarchy of blocks included in said each read-only snapshot copy of the production file, and wherein the deleting of the read-only snapshot copy of the production file includes keeping all descendants of said each block for which the read-only snapshot copy is not indicated as being an oldest snapshot copy of the production file including an identical version of said each block.

44. The method as claimed in claim 42, wherein the deleting of the read-only snapshot copy of the production file further includes keeping each block for which the read-only snapshot is indicated as being an oldest snapshot copy of the production file

including an identical version of said each block and a next-most recent snapshot copy of the production file is indicated as not being an oldest snapshot copy of the production file including an identical version of said each block.

45. The method as claimed in claim 44, which further includes, upon deleting the read-only snapshot copy of the production file, indicating that the next-most recent snapshot copy of the production file has become an oldest snapshot copy of the production file including an identical version of said each block for which the read-only snapshot is indicated as being an oldest snapshot copy of the production file including an identical version of said each block and a next-most recent snapshot copy of the production file is indicated as not being an oldest snapshot copy of the production file including an identical version of said each block.

46. The method as claimed in claim 42, wherein the deleting of the read-only snapshot copy of the production file includes deallocating each block for which the read-only snapshot copy is indicated as an oldest snapshot copy of the production file including an identical version of said each block and a next most recent snapshot copy of the production file is also indicated as an oldest snapshot copy of the production file including an identical version of a block corresponding to said each block, wherein said each block and the block corresponding to said each block are mapped to the same logical file addresses.

47. A method of operating a file server, the file server including storage containing a file system, and a processor coupled to the storage for accessing the file system, wherein the file system includes a production file, and snapshot copies of the production file, wherein the production file and the snapshot copies of the production file are organized as a version set including an inode for the production file, an inode for each snapshot copy of the production file, and a set of file blocks including data blocks and indirect blocks that are shared among the production file and the snapshot copies of the production file, wherein the method comprises:

the file server responding to a request to create a read-only snapshot copy of the production file, and when responding to the request to create a read-only snapshot copy of the production file, reserving for the production file a number of free file blocks of at least the number of blocks in the production file.

48. The method as claimed in claim 47, which includes the file server responding to a request to create a read-write snapshot copy of a specified file in the version set, and when responding to the request to create the read-write snapshot copy of the specified file in the version set, reserving for the read-write snapshot copy a number of free blocks of at least the number of blocks in the specified file in the version set.

49. The method as claimed in claim 47, which includes the file server responding to a request to restore the production file with a specified snapshot copy of the production file, and when responding to the request to restore the production file with the specified snapshot copy of the production file, by reserving for the production file a number of free

blocks of at least the difference between the number of blocks in the specified snapshot copy of the production file and the number of blocks in the production file.

50. A method of operating a file server, the file server including storage containing a file system, and a processor coupled to the storage for accessing the file system, wherein the file system includes a production file, and snapshot copies of the production file, wherein the production file and the snapshot copies of the production file are organized as a version set including an inode for the production file, an inode for each snapshot copy of the production file, and a set of file blocks including data blocks and indirect blocks that are shared among the production file and the snapshot copies of the production file, wherein the method includes:

the file server restoring the production file with a specified snapshot copy of the production file by responding to a request to prepare to restore the production file by preparing to restore the production file and reporting whether or not preparation is successful, and then responding a request to commit the preparation by restoring the production file with the specified snapshot copy of the production file.

51. The method as claimed in claim 50, which includes the file server preparing to restore the production file by creating a read-write snapshot copy of the specified snapshot copy of the production file, and which includes the file server committing the preparation by replacing the production file with the read-write snapshot copy of the specified snapshot copy of the production file and deleting the production file, so that the

read-write snapshot copy of the specified snapshot copy of the production file assumes the identity of the production file.

52. A method of operating a file server, the file server including storage containing a file system, and a processor coupled to the storage for accessing the file system, wherein the file system includes a production file, and snapshot copies of the production file, wherein the production file and the snapshot copies of the production file are organized as a version set including an inode for the production file, an inode for each snapshot copy of the production file, and a set of file blocks including data blocks and indirect blocks that are shared among the production file and the snapshot copies of the production file, wherein the method comprises:

the file server refreshing a specified snapshot copy of the production file by creating a new inode in the version set, copying contents of the inode of the specified snapshot copy into the new inode so that the new inode references blocks of the specified snapshot copy, using the inode of the specified snapshot copy to create a new snapshot copy of the production file by copying contents of the inode of the production file into the inode of the specified snapshot copy, and performing a file deletion upon the new node.

53. The method as claimed in claim 52, which includes the file server performing the file deletion upon the new inode asynchronously after using the inode of the specified read-only snapshot copy to create the new snapshot copy of the production file by copying contents of the inode of the production file into the inode of the specified snapshot copy.